Aufgabe 'Threads - Philosophenproblem'

> Beschreibung bei der Wikipedia: https://de.wikipedia.org/wiki/Philosophenproblem

Erstellen Sie bitte eine Anwendung in der das Philosophenproblem auftaucht. Beachten Sie die weiteren Aufgabenpunkte. Die auf der Seite beschriebene weitere Variante des Problems soll nicht nachprogrammiert werden.

- In Ihre Anwendung soll es mindestens 3 Philosophen geben.
- Jeder Philosoph hat einen eindeutigen Namen
- Optional. Gestalten Sie die Anwendung flexibel genug, um die Anzahl der Philosophen ohne viel Aufwand ändern zu können. Z.B.:

```
List<Philosoph> philosophen = getPhilosophen(3); //hier wird die Anzahl auf 3 gesetzt
...
```

Dabei können Sie einen Pool aus Philosophen im voraus erstellen.

- Simulieren Sie den Tagesablauf einzelner Philosophen mit entsprechenden Konsolenausgaben. Z.B.:

```
Sokrates denkt nach...
Sokrates hat Hunger
Sokrates nimmt die linke Gabel
Sokrates nimmt die rechte Gabel
Sokrates isst...
Sokrates legt die rechte Gabel ab
Sokrates legt die linke Gabel ab
```

- Der Tagesablauf eines Philosophen soll in einem Thread endlos wiederholt werden
- Überlegen Sie, auf welche Ressourcen die Philosophen gleichzeitig angewiesen sind. Synchronisieren Sie bitte den Zugriff auf die gemeinsamen Ressourcen so, dass es zu dem Deadlock kommen kann.

  Tipp: Versuchen Sie Deadlock wahrscheinlicher zu gestalten, indem die Threads an den Positionen künstrlich "gebremst" werden, wo das Anhalten zum Deadlock führt.

  Achtung! Die Aufgabe besteht nicht darin, das Philosophenproblem zu lösen! Es soll zum Deadlock kommen!
- Optional. Überlegen Sie, ob es möglich ist mit dem Lock-Konzept das Philosophenproblem zu lösen: http://docs.oracle.com/javase/8/docs/api/java/util/concurrent/locks/Lock.html